EXHIBIT 11

AV1 Technical Overview - A Summary

Mukul Shirvaikar, Ph.D.

Abstract

Video compression algorithms are a necessity for the implementation of video streaming and storage in any application, as it enables efficient transmission and storage of video data. Video compression techniques aim to reduce the amount of data required to represent a video signal while maintaining its visual quality. The following is a technical overview of the AV1 video encoder, created by the Alliance for Open Media, an open-source encoder first released in 2018. This includes a general summary of the syntax, reference frame system, coding blocks, and coding block operations.

1. Introduction

The AV1 (AOMedia Video 1) video encoder is an open-source tool released by the Alliance for Open Media in 2018. The goal of AV1 was to provide a free and open alternative to former de facto video encoders such as AVC and HEVC, both of which were released in 2013, and to build off Google's open-source VP9 encoder. Comparative analysis, such as Layek's [1] and Bender's [2] show comparable results to earlier video encoders, though at noticeably higher computational cost. Still, the open-source availability of AV1 makes it an attractive option for video applications. This paper aims to provide a simplified and condensed technical overview of the AV1 encoder, utilizing the IEEE's published overview as a basis [3].

2. Syntax Details

AV1 uses open bitstream units (or OBUs) to segment the bitstream into smaller information packets of varying length. A few examples of OBU types are:

- **Sequence Header**- Holds information about the bitstream sequence
- **Frame Header-** Holds information about the frame
- **Tile Group-** Holds a frame's tile data
- **Frame-** Frame header and tile data
- **Metadata** – Data about other data, can be anything from video time codes to author information
- **Tile List-** Similar to tile group with added data for tiles and reference frames

More syntax and information on discussed syntax can be found in [3], extensive list at [4].

3. Reference Frame System

Reference frames are chosen frames around the current frame being encoded used extensively in inter frame prediction (discussed later). AV1 carries up to 8 frames in the decoded frame buffer, 7 of which are reserved for reference frames. Which frames are used will be signaled explicitly in the frame header OBU. Generally, frames 1-4 are assigned to frames that come before the current frame and 5-7 to those after. The frames are signaled explicitly by the encoder in the bitstream, and it can use any of the 7 as reference frames. The encoder can choose which frame within the buffer to replace with the next frame

after frame encoding is complete. can bypass the buffer completely. There is also the option to use a frame stored in the buffer without displaying it, known as an Alternate Reference Frame. In order to display a frame, the codec can synthesize a new frame or use the frame stored directly in the buffer. Finally, the encoder also has the option to lower the resolution of the frame for compression and rescale the frame after, with some post-processing restoration.

4. Block Scheme

A segmentation of frames is necessary in order to break large images into less demanding pieces, which is exactly the function of AV1's block scheme. The largest block that AV1 can process is defined as a superblock, which can be 128x128 or 64x64 luma samples. These superblocks make up tiles, which are rectangular arrays consisting of anywhere from one superblock to 4096x2304 luma samples (which comprise the superblocks). The tiles can be uniform (equal width and height to one another) or non-uniform (variable width and height, but still must meet at corners with one another), which is useful for allowing more computation on complex areas by allocating them a larger number of smaller superblocks.

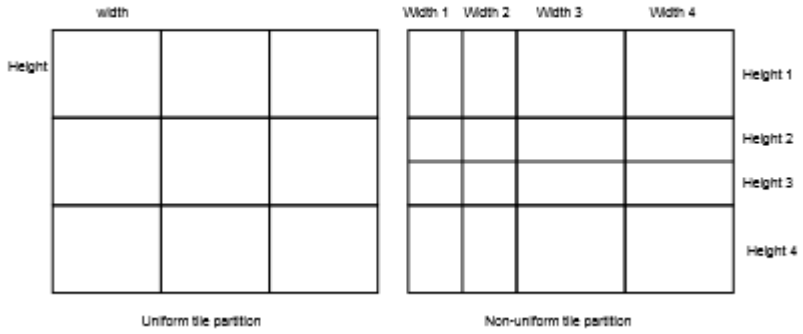

Figure 1: Uniform and Non-uniform Block Schemes [3]

5. Coding Block Operations

Coding blocks can be further subdivided and operated upon, most notably for prediction that speeds up the decoding process and for compression utilizing transform coding and quantization.

A. Coding Block Partitioning:

The essence of the AV1 coding block partition is that each superblock can be processed as-is or recursively partitioned down to a minimum of 4x4 luma samples. Each block can be divided in 10 separate ways, but only by dividing it into fourths can the codec continue the division further, treating each fourth as its own block. A special process must be used to encode 4x4 luma samples to avoid latency issues, which is further discussed in [3].

B. Intra Frame Prediction:

Intra frame prediction is used when decoding a frame. It's job is to predict pixel values by using other (already decoded) pixel information elsewhere in the frame.

i) ***Directional:***

This mode allows prediction of a pixel by other pixels in the frame, choosing from 8 directions (down/left and up/right with 45° steps between) as base with 3 possible 3° steps in either direction. If the prediction points to a sub-pixel position, an interpolation filter is used to generate a pixel.

ii) ***Non-Directional Smooth:***

Used within a coding block with left column and top row as reference pixels for interpolation (filling bottom row and right column predictions for a closed boundary). The four prediction modes are:

- SMOOTH_H_PRED: Predicts along a horizontal line from reference weighted by distance from reference
- SMOOTH_V_PRED: Predicts along a vertical line from reference weighted by distance from reference
- SMOOTH_PRED: (SMOOTH_H_PRED+SMOOTH_V_PRED)/2
- PAETH_PRED: Uses top and left boundary to create a prediction

iii) ***Recursive Intra Prediction:***

This mode runs through a coding block, taking 4x2 blocks of pixels, and predicts their values by multiplying the values of the pixels above, to the left, and diagonally to the top left by a determined parameter and adding the results.

iv) ***Chroma from Luma Prediction:***

This mode works in a relatively self-explanatory way, multiplying a luma sample's value by a set factor in order to derive a predicted chroma value [5].

v) ***Intra block copy:***

This function is used for intra-block motion compensation by copying a smaller block of pixels to a new location within the coding block. An important note is that it cannot be used with post-processing filters, which may affect the quality of some videos [6].

vi) ***Color palette:***

The color of each pixel is assigned to a base value and entropy coded. The number of base values can range from 2 to 8, with less meaning higher compression with lower fidelity and vice-versa.

C. Inter Frame Prediction:

Inter Frame prediction in similar to methods discussed above, but it utilizes reference frames before and after the current frame, making it a useful tool for motion compensation.

i) Translational motion compensation:

A vector is generated by the codec (sum of absolute differences and sum of squared

error are the most common prediction methods) that predicts the displacement of a pixel between frames. The pixel is then "moved" (its value copied) to the new location with 1/8th pixel accuracy by using interpolation filters. Three different interpolation filters of different qualities (SMOOTH, REGULAR, and SHARP) can be selected for both the horizontal and vertical directions.

ii) ***Affine Model Parameters and Motion Compensation:***

Affine prediction can be used for scaling, rotation, and translation of an image, making a versatile tool for . This is done by creating a matrix of values that controls each of the three transformations and using matrix multiplication to get values for horizontal and vertical offsets of a given pixel between frames. The derivation of the matrix of affine parameters is discussed in-depth in [3].

iii) Compound Predictions:

Compound predictions define multiple separate ways to linearly combine two motion compensation predictions. These modes are: Distance Weighted Predictor, Average Predictor, Difference Weighted Predictor, Wedge Mode, Overlapped Block Motion Compensation, and Compound Inter-Intra Predictor. The best scenarios for the usage of each one is discussed further in [3].

Fig. 13: Illustration of the compound prediction modes. The distance weighted predictor uniformly combines the two reference blocks. The difference weighted predictor combines the pixels when their values are close (e.g., the dial plate and the numbers), and picks one reference when the difference is large (e.g. the clock hands). The wedge predictor uses one of the preset masks to split the block into two sections, each filled with one reference block's pixels. In the above example, it stitches the lower part predictor 1 and the higher upper part of predictor 2 as the compound prediction.

Figure 2: Illustration of Each Compound Prediction Mode [3]

D. Dynamic Motion Vector Referencing Scheme:

The Dynamic Motion Vector Referencing Scheme is one of the more complex systems within AV1 indented to decrease computational complexity by storing vectors more efficiently. Instead of

storing vectors directly, AV1 compares spatial neighbors and temporal neighbors using a vector search for neighboring blocks and a motion field for reference frames with vectors that cross the current frame. The vectors are ranked by appearance counts within the search range, and the encoder will select the closest eligible vector as a reference vector. The encoder then stores the reference vector's index, and the motion vector difference (from the reference) is entropy coded. Overall, higher efficiency is achieved by storing vectors as differences of more "popular" vectors that are helpful in entropy coding.

E. Transform Coding:

The general idea of transform coding is to encode pixel values as a composite of frequencies. Using some function-based transform. The coefficients of each frequency for each dimension can then be stored in a much smaller array.

i) Transform Block Size:

First, coding blocks can be further divided into smaller pieces with a maximum level 2 recursive partition. The maximum block size is 64x64 luma samples and the minimum is 4x4 luma samples.

ii) Transform Kernels:

AV1 allows each block to independently select from 4 1-D transform kernels for a total of 16 2-D transform kernels. The four 1-D kernels are DCT, ADST, Flipped ADST, and Identity transform. The kernel used for each dimension is decided through a myriad of factors discussed further in [3].

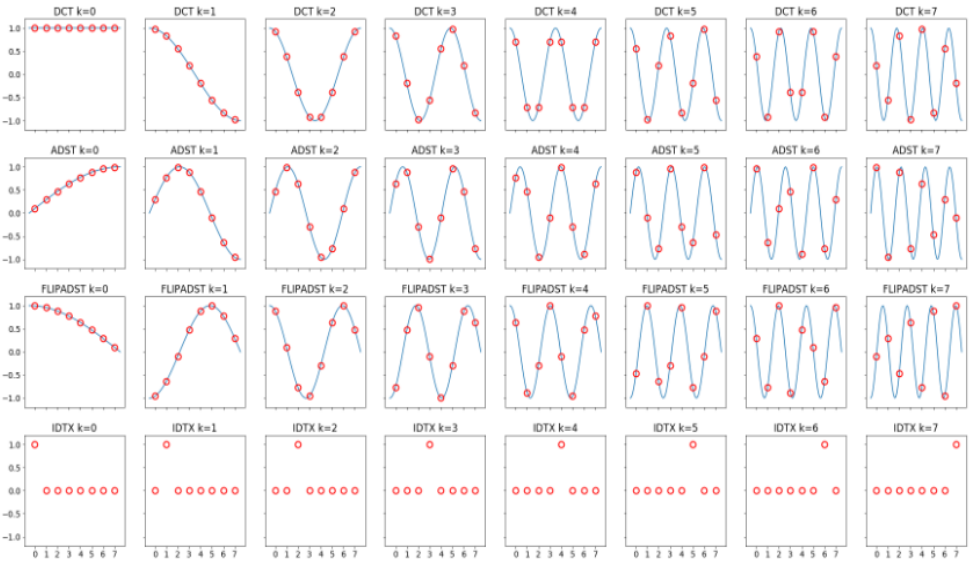

Fig. 24: Transform kernels of DCT, ADST, FLIPADST and IDTX for dimension $N = 8$. The discrete basis values are displayed as red circles, with blue lines indicating the associated sinusoidal function. The bases of DCT and ADST (a variant with a fast butterfly structured implementation) take the form of $cos(\frac{(2n+1)k\pi}{2N})$ and $sin(\frac{(2n+1)(2k+1)\pi}{4n})$ respectively, where n and k denote time index and the frequency index, taking values from $\{0, 1, ..., N-1\}$. FLIPADST utilizes the reversed ADST bases, and IDTX denotes the identity transformation.

Figure 3: Kernel Waveforms [3]

F. Quantization:

The transform coefficients found in section E are divided by some constant and rounded to an integer (quantized) for further compression, and the decoder will multiply by said constant to estimate the true values. The constant is determined by the quantization parameter in AV1, which ranges from 0 to 255, with 0 being lossless and 255 being maximum compression. AV1 supports multiple ways of quantizing the coefficients, with 15 different matrices of quantization weights. As always, further analysis is provided in [3].

6. Conclusion:

AV1 is a very versatile codec and an extremely attractive option for implementing in video applications. It may be computationally complex, but it is a new codec with continuing support, transparent design, and an in-depth toolkit. Further research on AV1 is still being conducted, and further in-depth discussion of its technical aspects has been published by IEEE [3] with discussion of its entropy coding system, post-processing, and performance, which were not covered by this paper, will be covered in the next version.

7. References

1. M. A. Layek et al., "Performance analysis of H.264, H.265, VP9 and AV1 video encoders," 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), Seoul, Korea (South), 2017, pp. 322-325, doi: 10.1109/APNOMS.2017.8094162.
2. I. Bender, D. Palomino, L. Agostini, G. Correa and M. Porto, "Compression Efficiency and Computational Cost Comparison between AV1 and HEVC Encoders," 2019 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2019, pp. 1-5, doi: 10.23919/EUSIPCO.2019.8903006.
3. J. Han et al., "A Technical Overview of AV1," in Proceedings of the IEEE, vol. 109, no. 9, pp. 1435-1462, Sept. 2021, doi: 10.1109/JPROC.2021.3058584.
4. P. de Rivaz and J. Haughton, “AV1 bitstream & decoding process specification.” [Online]. Available: https://aomediacodec.github. io/av1-spec/av1-spec.pdf
5. J. Jeong, G. Gankhuyag and Y. -H. Kim, "Fast Chroma Prediction Mode Decision based on Luma Prediction Mode for AV1 Intra Coding," 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2019, pp. 1050-1052, doi: 10.1109/ICTC46691.2019.8939936.
6. J. Li et al., "Intra Block Copy for Screen Content in the Emerging AV1 Video Codec," 2018 Data Compression Conference, Snowbird, UT, USA, 2018, pp. 355-364, doi: 10.1109/DCC.2018.00044.